Tutorial

omicR

Windows and Linux

Berenice Talamantes Becerra

2-25-2021

Contents

Figures

Introduction

omicR creates fasta files, downloads genomes from NCBI using the refseq number, creates databases to run BLAST+, runs BLAST+ and filters these results to obtain the best match per sequence.

These scripts can be used to run BLAST alignment of short-read (DArTseq data) and long-read sequences (Illumina, PacBio… etc). You can use reference genomes from NCBI, genomes from your private collection, contigs, scaffolds or any other genetic sequence that you would like to use as reference.

You can skip this tutorial and watch the tutorial video in YouTube

Installing omicR on Windows (1:30 min)

https://youtu.be/19zn7WoKtbg

Using omicR with Graphical User Interface (~20 min)

https://youtu.be/pdMio2vj-FM

Using omicR in R Studio (~20 min)

https://youtu.be/g46a3Gd1q6g

Installing omicR with graphical user interface

Windows

Requirements:

BLAST+ latest version: https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/

Installation:

There are 2 options to run omicR in windows.

1) **Option 1.**
   Download the code from GitHub:
   https://github.com/BTalamantesBecerra/omicR_for_Windows.
   Download the zip directory "omicR.zip". Unzip this directory and double click the executable file "omicR.exe" with the image of the green parrot. This will open a window where you can run the scripts. You do not need to install anything as everything is compiled into this file and you can start running your analysis. You can see this step in this video: https://youtu.be/19zn7WoKtbg

2) **Option2.** If you cannot open the executable file, you may need to run the script directly through Python. For this you need to install the following:
   Download the code from GitHub:
   https://github.com/BTalamantesBecerra/omicR.

   a. Python V3 or latest: https://www.python.org/downloads/
   b. Biopython https://biopython.org/

   -Open the script "omicR.py" in Python and run it.
   -This will open a Window where you can run the scripts.

Linux

If you are using Linux, it is likely that Python is already installed. Download the code from GitHub: https://github.com/BTalamantesBecerra/omicR. For Linux, download the code from https://github.com/BTalamantesBecerra/omicR_for_linux

Requirements:

a. Python V3 or latest: https://www.python.org/downloads/
    i. Module: tkinter
b. Biopython https://biopython.org/
    i. Module: entrez
c. BLAST+ latest version: https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/

-Open the script "omicR.py" in Python and run it.
-This will open a window where you can run the scripts.

Remember to add Python, Biopython and BLAST into your System Variables Path. As general practice, avoid installing your software in directories such as "C:\\Program files\" as the space between words will cause problems.

Running “omicR” with graphical user interface

Before you run a test, download the Sample data. The csv file is called “SampleData_Enterococcus_faecium.csv”. **Only CSV files are accepted as input for these scripts.**

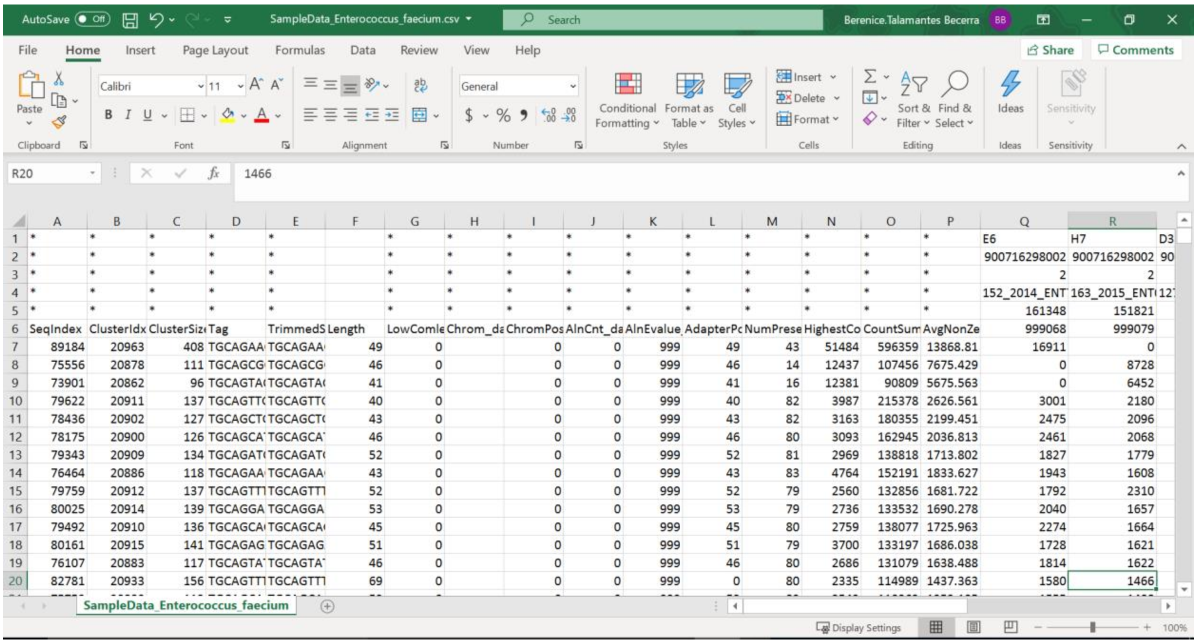


Figure 1 Sample Data of E. faecium

Open or run the “omicR” executable. The window should look like this. As you can see, each button runs a different script.

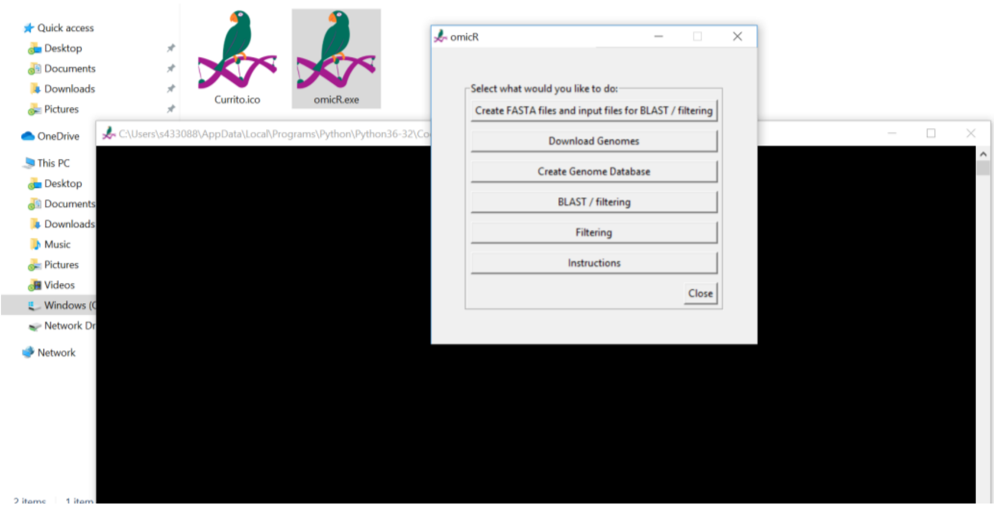


Figure 2 First windows of omicR GUI.

STEPS

1. **Create FASTA files and input files for BLAST / filtering.**
   This will open another window. Select the paths to the Sample Data file provided and select parameters as required.

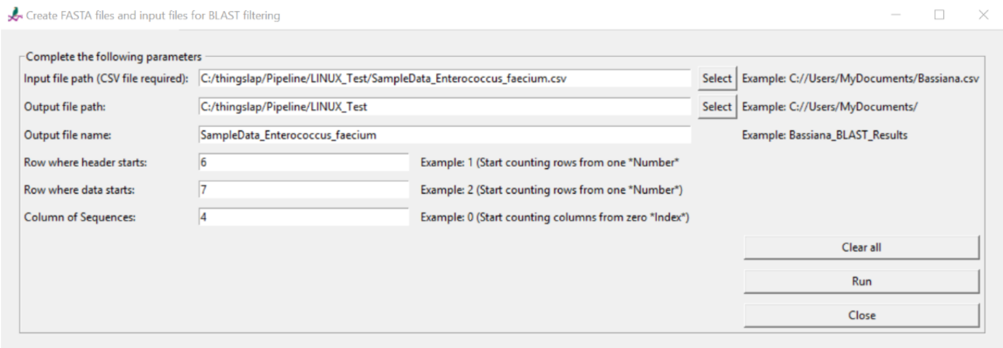


Figure 3 Create Fasta files and input files for BLAST using GUI.

The row where header starts is row 6 and the row where data starts is 7. And in this case, for this file we need the Trimmed Sequences column, it is in column 4. For columns you need to start counting from 0.

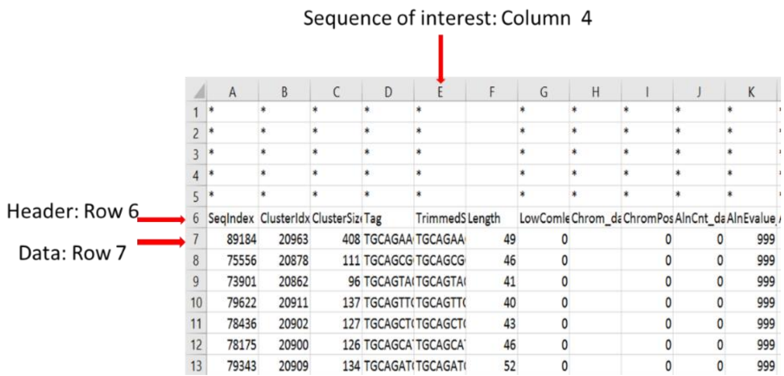


Figure 4 Display of how rows and headers should be selected.

REMEMBER TO CHECK THE HEADER AND DATA OF YOUR FILE BEFORE RUNNING ANY SCRIPT

This will create 2 files, one Fasta file and one copy of your original file including an extra column containing a Unique ID. These files are required for the following steps.

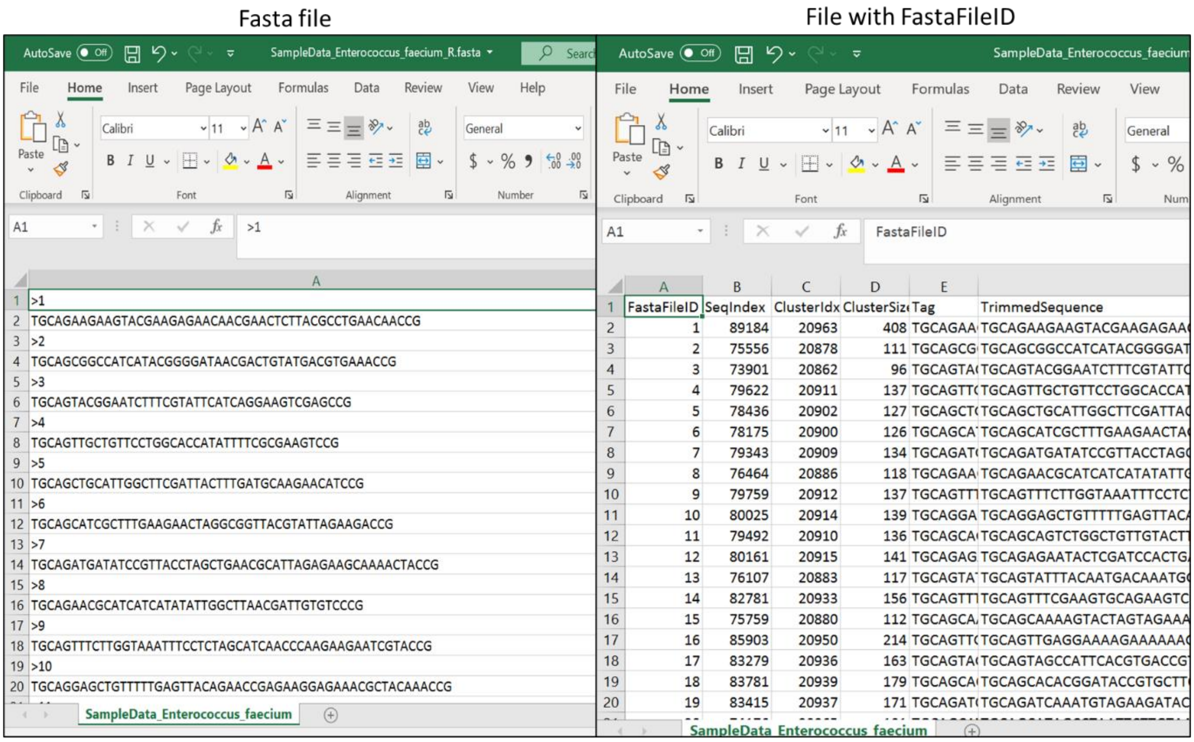


Figure 5 Output files generated from script "Create Fasta files".

2. **Download genomes from the NCBI.**

   For this example, we need the genome and plasmid of *E. faecium* (https://www.ncbi.nlm.nih.gov/assembly/GCF_010120755.1). The RefSeq numbers needed for this tutorial are: NZ_CP039729.1, NZ_CP039730.1

   To run the script, write your email, select the RefSeq accession numbers, select the output path and name for the genome.

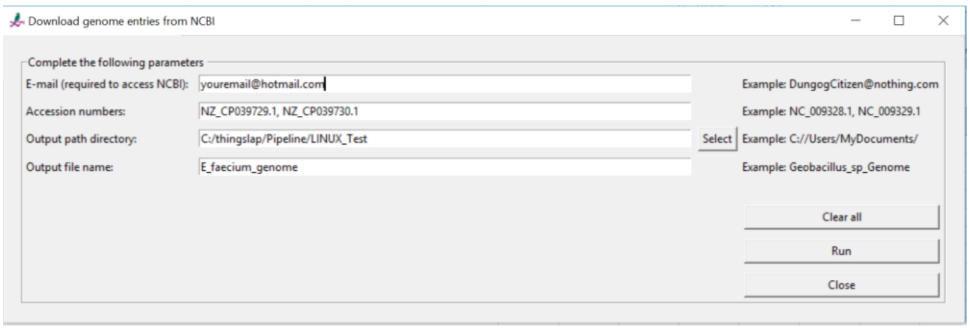


Figure 6 Download genome entries from NCBI.

This will create a directory with the name given to the genome.

Please note that this option is suitable for small genomes or small chromosomes. To download larger genomes, it is recommended to use the internet browser option. This example of *E. faecium* should download in less than 5 minutes. If you use this method to download the Chicken genome it can take up to 5 hours.

3. Create NCBI database for BLAST+.

To create the database for BLAST, select the path to the “bin” directory where BLAST+ was installed.

Select the location of the genome/sequences/scaffolts/contigs or whatever you would like to use as reference to BLAST, then select the type of database. For this example we are working with nucleotides, so we select “nucl”.

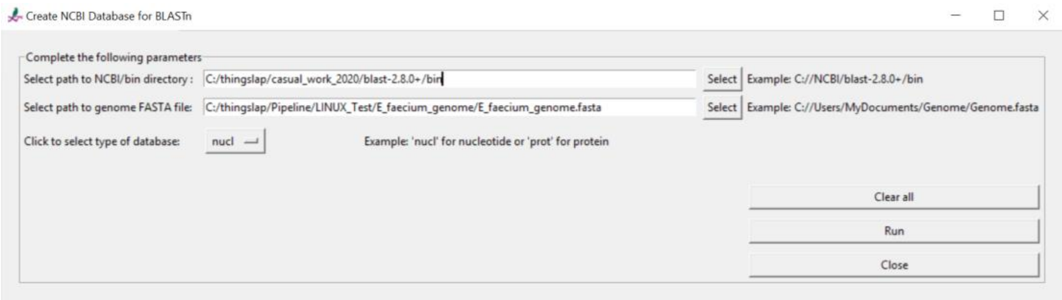
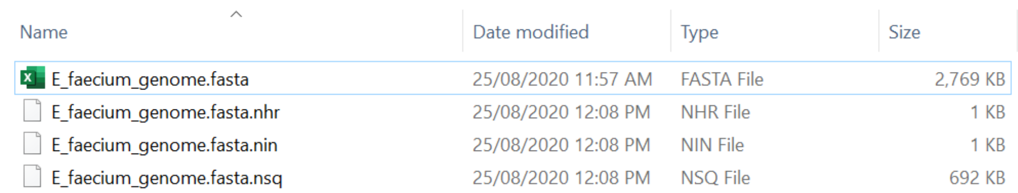


Figure 7 Create NCBI genome database for BLAST+.

The script will create 3 files in the same location as your database, with the same name of the reference and the terminations “.nhr”, “nin”, “nsq”.



Name	Date modified	Type	Size
E_faecium_genome.fasta	25/08/2020 11:57 AM	FASTA File	2,769 KB
E_faecium_genome.fasta.nhr	25/08/2020 12:08 PM	NHR File	1 KB
E_faecium_genome.fasta.nin	25/08/2020 12:08 PM	NIN File	1 KB
E_faecium_genome.fasta.nsq	25/08/2020 12:08 PM	NSQ File	692 KB

Figure 8 Example of files created after creating BLAST database.

4. BLAST and filtering.

To run the BLAST analysis, you need to select the path to the bin directory where you installed BLAST+, then select your path to the database created in the previous step, the output path, the output file name, the output path to the file with Unique ID and BLAST parameters.

NOTES:

- If you only used a fasta file as input, and you do not have the file with Unique ID, you can still run this script.
- If you are running a BLAST alignment of similar sequences, for example Turtle Genome Vs Turtle Sequences, the recommended parameters are: Word Size 11, Percentage identity 70, Number of threads 4, Output format 6, Percentage Overlap 0.8, bitscore 50.
- If you are running a BLAST of highly dissimilar sequences because you are probably looking for sex linked hits in a distantly related species, and you are aligning sequences of Chicken Genome Vs Bassiana, use a Percentage overlap of 0.01, Bitscore of 30 and tick the option of "Discontinuous Mega BLAST"

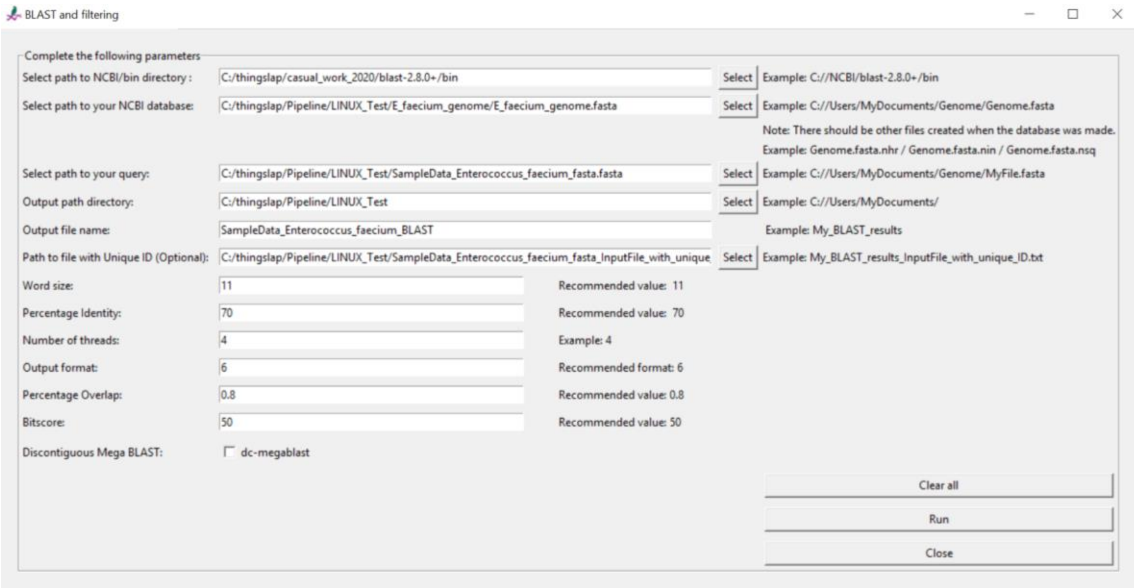


Figure 9 BLAST and filtering example.

This step takes less than 5 minutes. It will produce 5 files, or 3 files if you didn't provide the File with UniqueID.

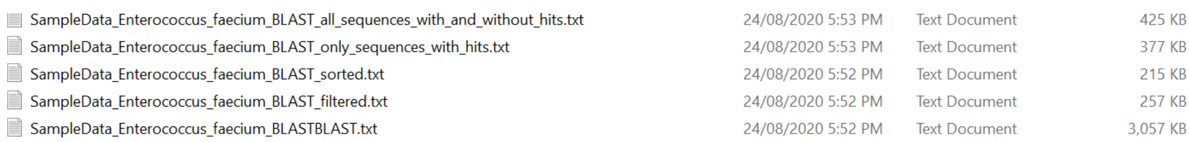


Figure 10 Example of files produced after running the BLAST and filtering script.

Description of files produced.

- **File 1. SampleData_Enterococcus_faecium_BLASTBLAST.txt**
  This is the raw BLAST output. This file does not contain any headers and it is not filtered.

- **File 2. SampleData_Enterococcus_faecium_BLAST_filtered.txt**

This file has headers and an extra column with the percentage overlap. Default filtering parameters for this tutorial are: Percentage Overlap >80%, bitscore >50, Percentage Identity>70. The percentage overlap can be modified according to the BLAST results expected. This file may contain multiple hits per sequence.

The BLAST output is formatted as a table using output format 6, with columns defined in the following order: " qseqid sacc stitle qseq sseq nident mismatch pident length evalue bitscore qstart qend sstart send gapopen gaps qlen slen". These are:

- qseqid: query (e.g., unknown gene) sequence id
- sacc: Subject accession
- stitle: Subject Title
- qseq: Aligned part of query sequence
- sseq: Aligned part of subject sequence
- nident: Number of identical matches
- mismatch: number of mismatches
- pident: percentage of identical matches
- length: alignment length (sequence overlap)
- evalue: expect value
- bitscore: bit score
- qstart: start of alignment in query
- qend: end of alignment in query
- sstart: start of alignment in subject
- send: end of alignment in subject
- gapopen: number of gap openings
- gaps: Total number of gaps
- qlen: Query sequence length
- slen: Subject sequence length

- **File 3. SampleData_Enterococcus_faecium_BLAST_sorted.txt**
  This file contains only one hit per sequence. The best match will be selected by considering the following values ranked in order. First considering the highest percentage identity, then the highest Percentage overlap, then the highest bitscore. Only one Query per sequence is kept based on these selection criteria.
  If you did not provide the file with UniqueID, this filtered and sorted BLAST output file will be your final result.

- **File 4. SampleData_Enterococcus_faecium_BLAST_only_sequences_with_hits.txt**
  This file uses the UniqueID assigned to each sequence and writes the BLAST results back into the original file. This file only contains sequences that had a BLAST hit to something in the reference.

- **File 5. SampleData_Enterococcus_faecium_BLAST_all_sequences_with_and_without_hits.txt**

This file uses the UniqueID assigned to each sequence and writes the BLAST results back into the original file. This file contains all sequences, including those with and without hits, written back into the original file.

5. **Additional filtering.**

If you would like to run additional filtering without re-running the BLAST, you can use the BLAST result obtained in the previous step as an input and filter again with different parameters. For example, using a higher or lower percentage overlap or bitscore.

Note: This script only takes input files with BLAST Tabular output format 6 with the ordered set of columns described in the previous step.

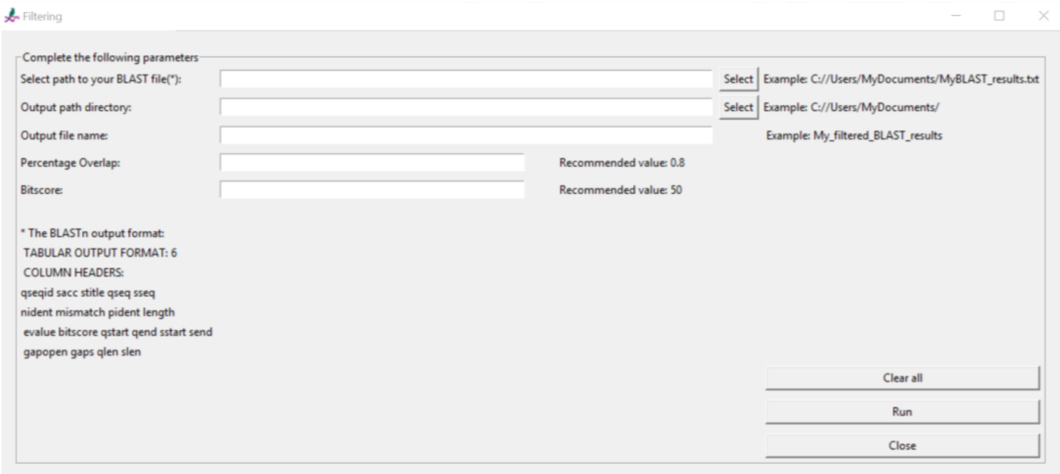


Figure 11 Additional filtering window.

omicR for R Studio

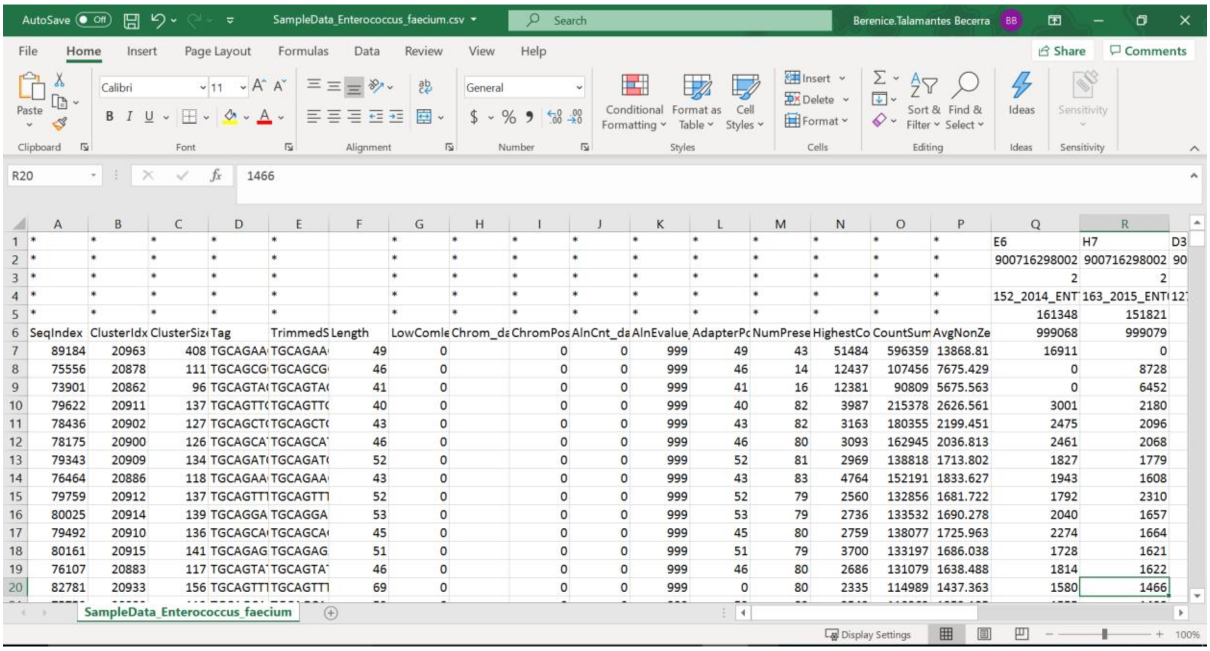
omicR for R studio runs using Python scripts through R Studio. You can download the R project in: https://github.com/BTalamantesBecerra/omicR_for_RStudio

1) You need to install the following:
    a. Python V3 or latest: https://www.python.org/downloads/
    b. Biopython https://biopython.org/
    c. Download the omicR code. This should include at least the following:
       "omicR.Rproj",
       "TestingPyCharm_MKfasta.py",
       "TestingPyCharm_Downloading_genomes.py",
       "TestingPyCharm_MakeDataBase.py",
       "TestingPyCharm_BLAST_filtering_and_all.py",
       "TestingPyCharm_NCBI_BLAST_filtering.py",

Running “omicR” in R studio

Before you run a test, download the Sample data. The csv file is called “SampleData_Enterococcus_faecium.csv”.

Note: The pipeline only takes csv files as input.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	*	*	*	*	*		*	*	*	*	*	*	*	*	*	*	E6	H7	D3
2	*	*	*	*	*		*	*	*	*	*	*	*	*	*	*	900716298002	900716298002	90
3	*	*	*	*	*		*	*	*	*	*	*	*	*	*	*	2	2	
4	*	*	*	*	*		*	*	*	*	*	*	*	*	*	*	152_2014_ENT	163_2015_ENT	12
5	*	*	*	*	*		*	*	*	*	*	*	*	*	*	*	161348	151821	
6	SeqIndex	ClusterIdx	ClusterSiz	Tag	TrimmedS	Length	LowComle	Chrom_da	ChromPos	AlnCnt_da	AlnEvalue	AdapterPc	NumPrese	HighestCo	CountSum	AvgNonZe	999068	999079	
7	89184	20963	408	TGCAGAA	TGCAGAA	49	0		0	0	999	49	43	51484	596359	13868.81	16911	0	
8	75556	20878	111	TGCAGCG	TGCAGCG	46	0		0	0	999	46	14	12437	107456	7675.429	0	8728	
9	73901	20862	96	TGCAGTA	TGCAGTA	41	0		0	0	999	41	16	12381	90809	5675.563	0	6452	
10	79622	20911	137	TGCAGTT	TGCAGTT	40	0		0	0	999	40	82	3987	215378	2626.561	3001	2180	
11	78436	20902	127	TGCAGCT	TGCAGCT	43	0		0	0	999	43	82	3163	180355	2199.451	2475	2096	
12	78175	20900	126	TGCAGCA	TGCAGCA	46	0		0	0	999	46	80	3093	162945	2036.813	2461	2068	
13	79343	20909	134	TGCAGAT	TGCAGAT	52	0		0	0	999	52	81	2969	138818	1713.802	1827	1779	
14	76464	20886	118	TGCAGAA	TGCAGAA	43	0		0	0	999	43	83	4764	152191	1833.627	1943	1608	
15	79759	20912	137	TGCAGTT	TGCAGTT	52	0		0	0	999	52	79	2560	132856	1681.722	1792	2310	
16	80025	20914	139	TGCAGGA	TGCAGGA	53	0		0	0	999	53	79	2736	133532	1690.278	2040	1657	
17	79492	20910	136	TGCAGCA	TGCAGCA	45	0		0	0	999	45	80	2759	138077	1725.963	2274	1664	
18	80161	20915	141	TGCAGAG	TGCAGAG	51	0		0	0	999	51	79	3700	133197	1686.038	1728	1621	
19	76107	20883	117	TGCAGTA	TGCAGTA	46	0		0	0	999	46	80	2686	131079	1638.488	1814	1622	
20	82781	20933	156	TGCAGTT	TGCAGTT	69	0		0	0	999	0	80	2335	114989	1437.363	1580	1466	

Figure 12 Example of CSV input file.

STEPS

1. **Create FASTA files and input files for BLAST / filtering.**
    - Select the script “mkfastafile.R”.
    - Clean the global environment before running these scripts.

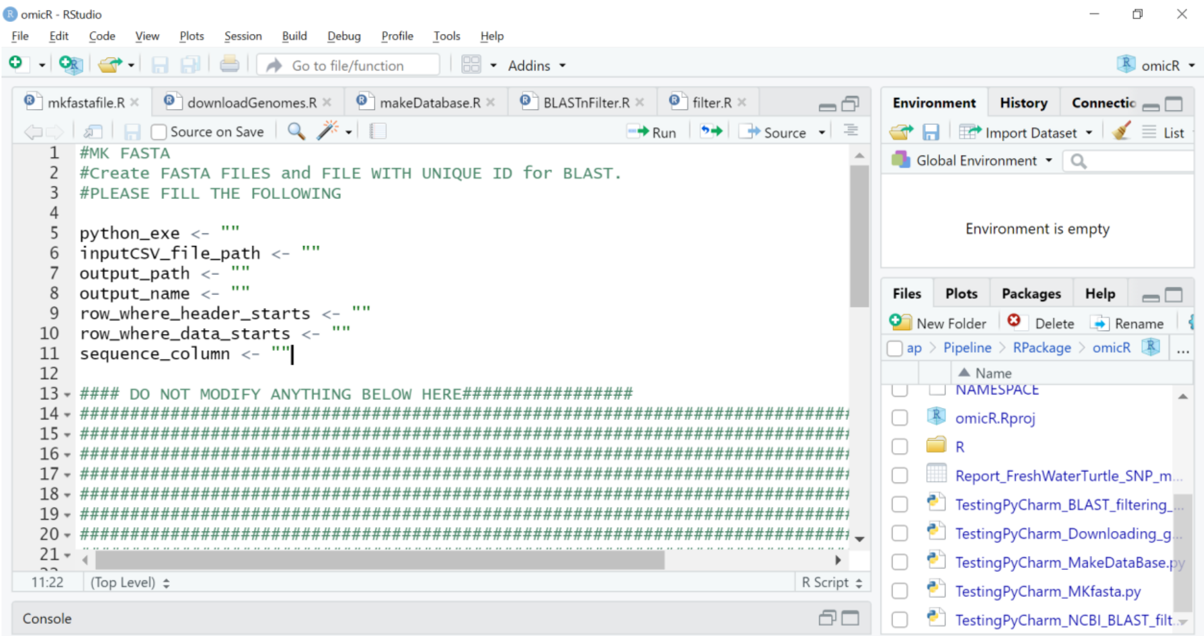


Figure 13 mkfastafile.R script before adding details.

- Now write the paths to your files as indicated. Do not modify anything below the sign “DO NOT MODIFY ANYTHING BELOW HERE”.

Notes:

- ➢ Remember to use double backslashes “\\” to allow the script to run successfully.
- ➢ After you fill the parameters, it should look like this:

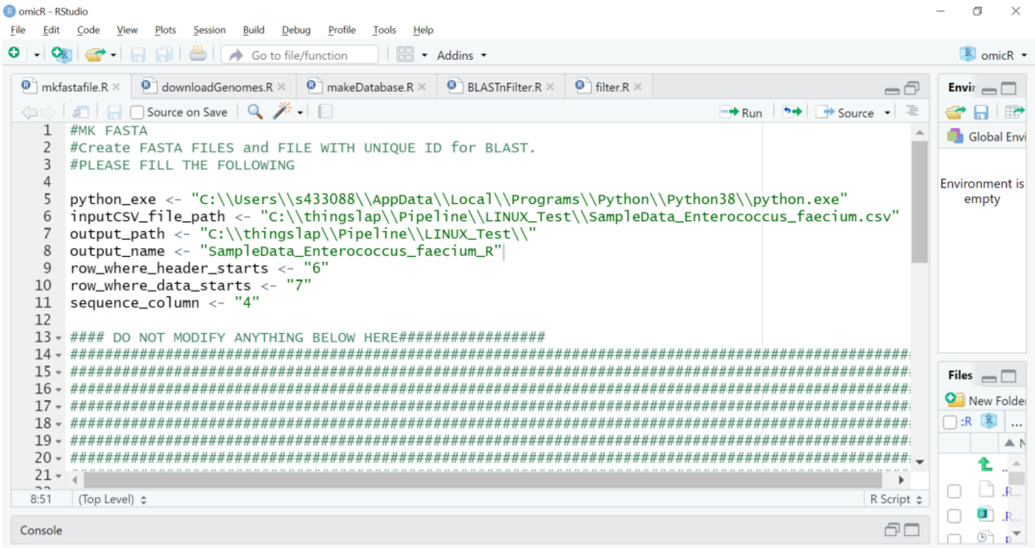


Figure 14 mkfastafile.R script after adding details.

- Press CTRL+A to select the entire code

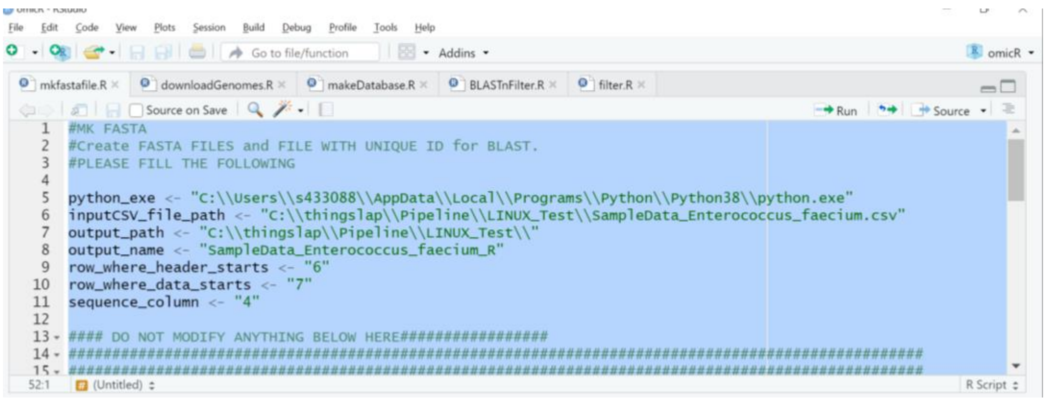


Figure 15 mkfastafile.R script after selecting all the code.

- Run the code

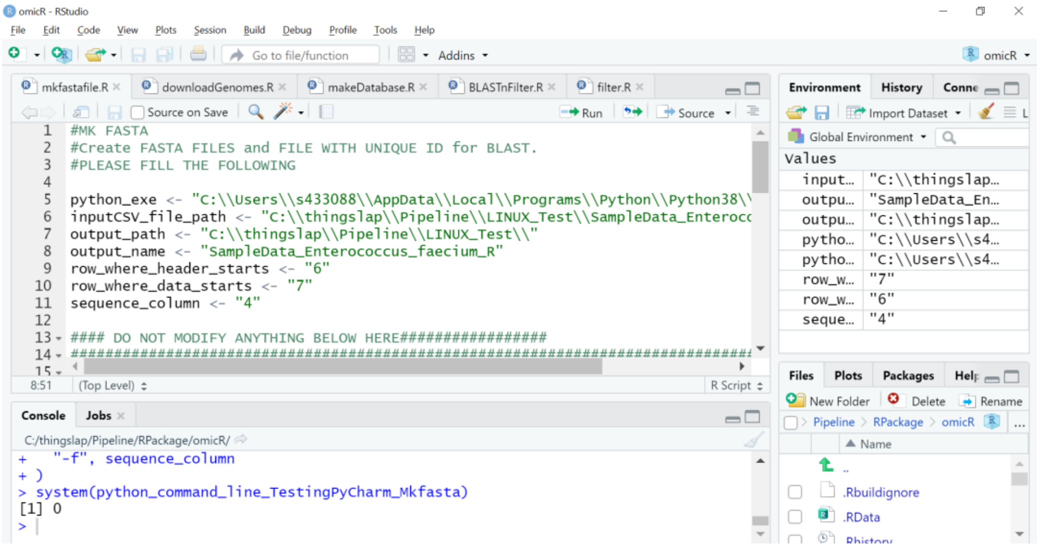


Figure 16 mkfastafile.R script after running the code.

This code will produce two files that are needed for the following steps.

Fasta file | File with FastaFileID

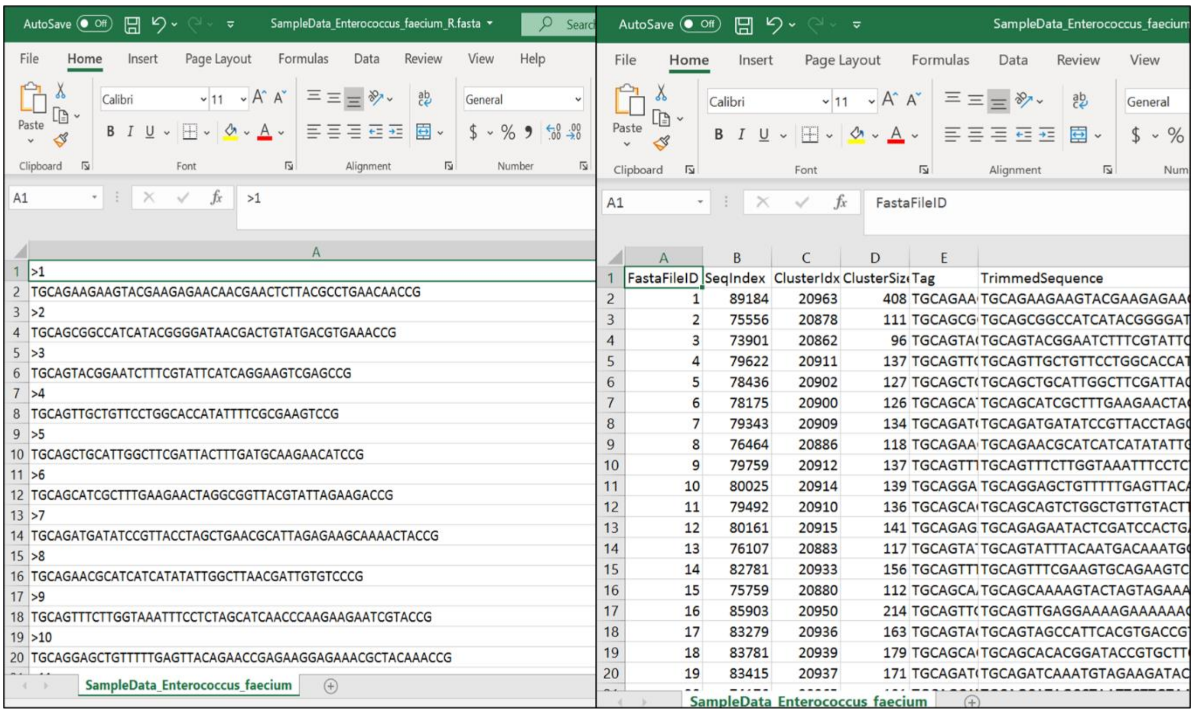


Figure 17 Example of fasta file and file with Unique ID.

2. Download genomes from the NCBI.

- Select the script “donwnloadGenomes.R”.
- Clean your global environment before running this script.

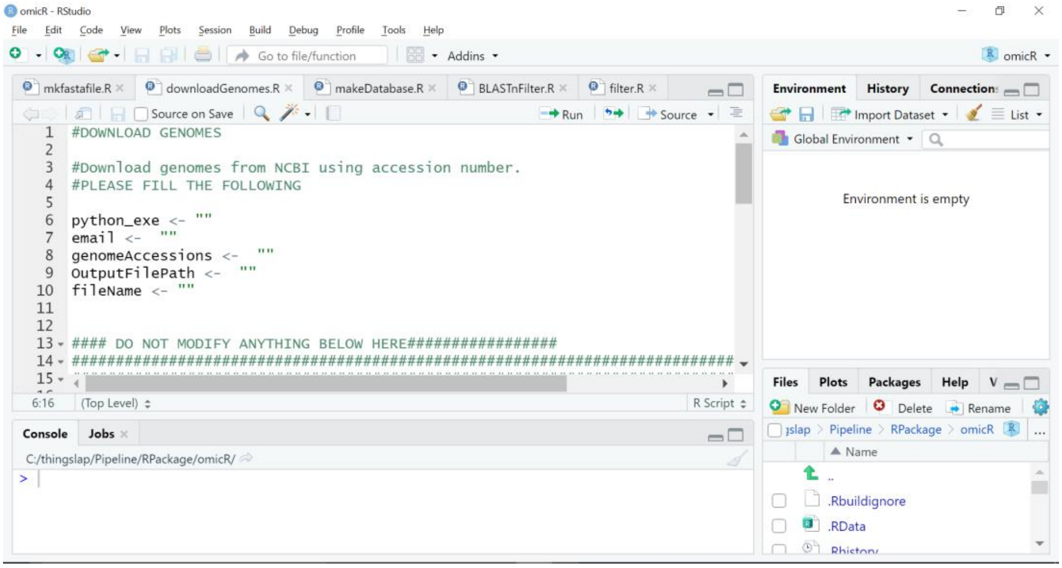


Figure 18 Example of donwnloadGenomes.R script.

- Fill the scripts with the path to the python executable files, your email, the RefSeq numbers separated by a comma and **WITHOUT SPACE** IN BETWEEN, the output file path and name.
- Remember to use \\ in all your paths.

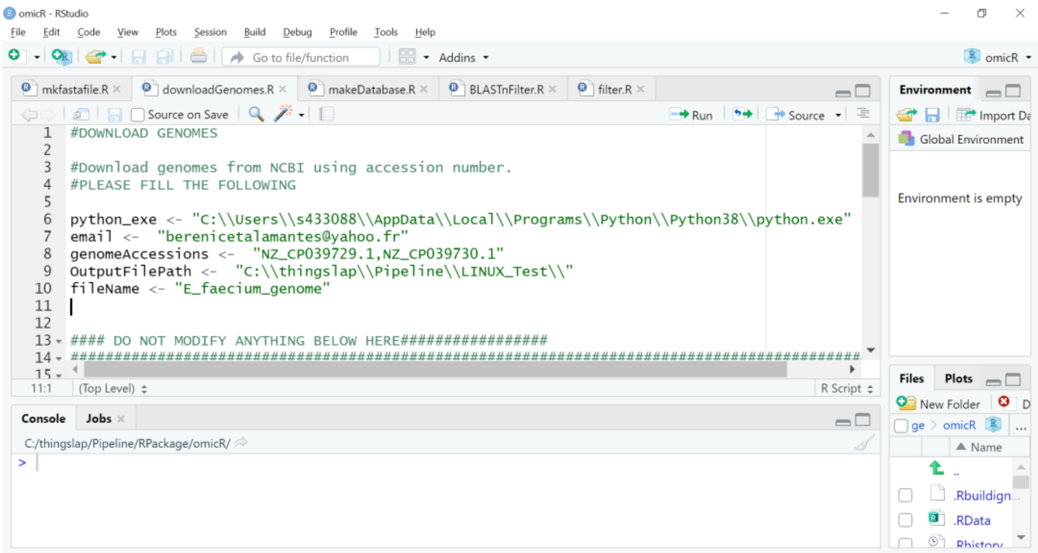


Figure 19 Example of donwnloadGenomes.R script after adding parameters.

- Save your script, select all the code and run.

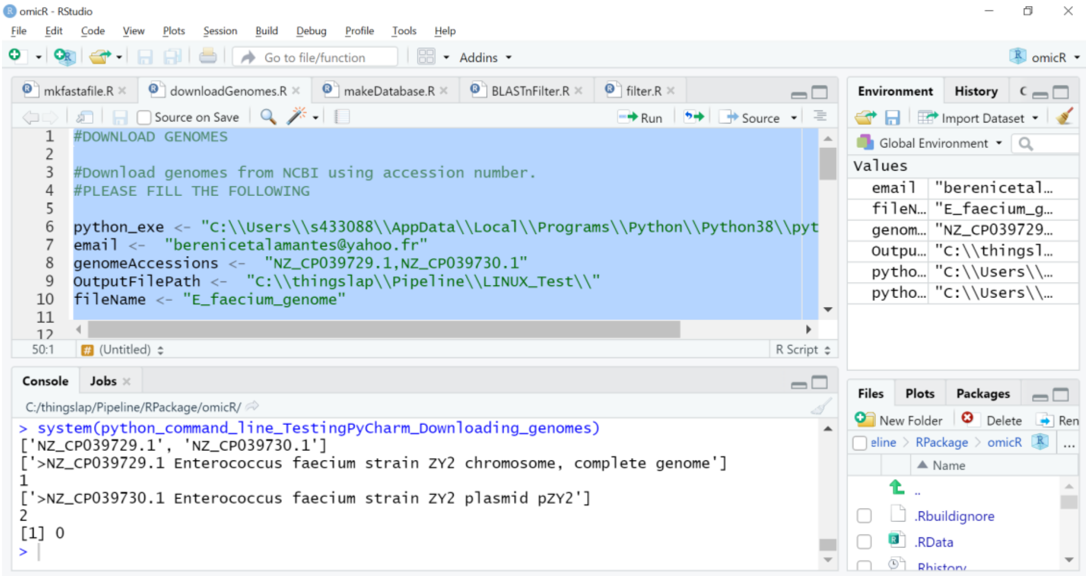


Figure 20 Example of donwnloadGenomes.R script after running the code.

This step creates a directory with a single fasta file that includes all the RefSeq numbers fetched.

3. **Create BLAST+ database.**

- Select the script “makeDatabase.R” and complete the script by typing the paths for your computer.
- Clean your global environment before running this script.

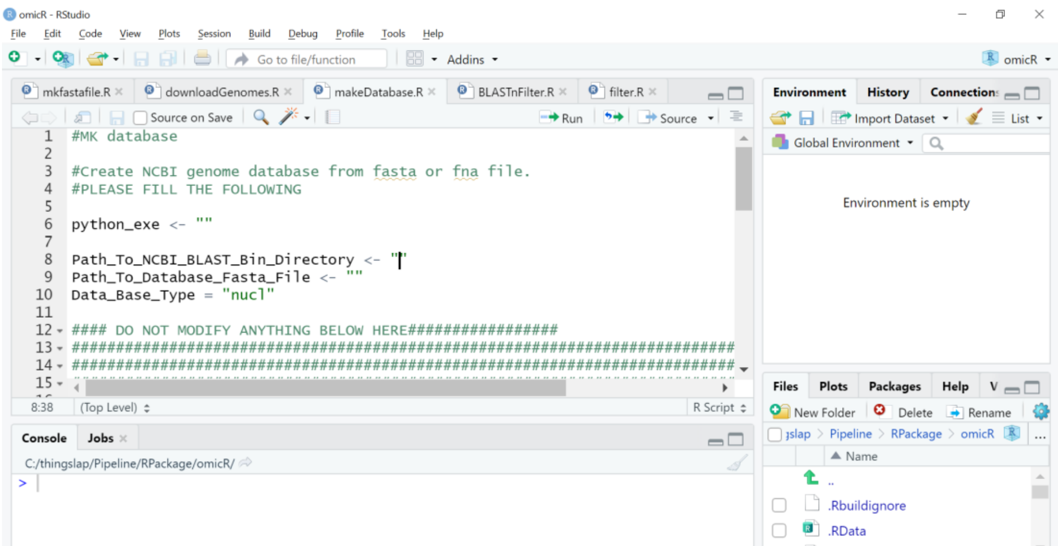


Figure 21 Example of window showing “makeDatabase.R”

- Fill the script and add the path to the Python executable, path to the BLAST+ bin directory, path to the file (genome, contigs, scaffolds…etc) to turn into database, and the type of database (for nucleotides select “nucl”).

➢ Remember to type \\ in the path.

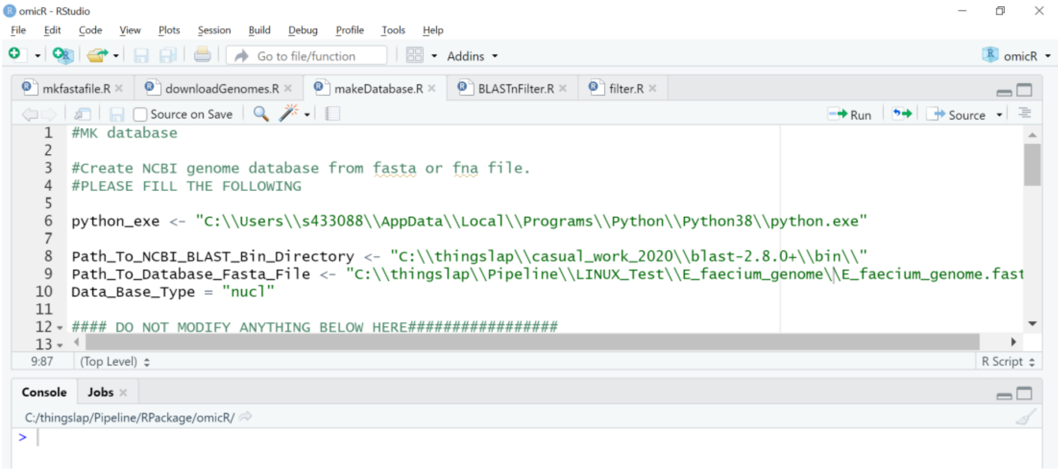


Figure 22 Example of script “makeDatabase.R” after filling parameters.

- Save the script, select all the code and run it.

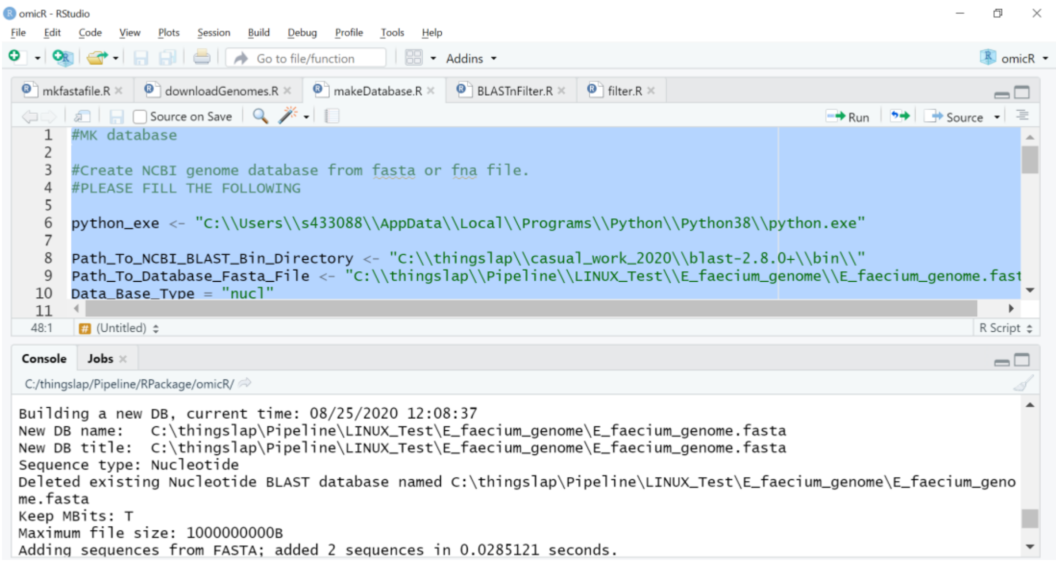


Figure 23 Example of script “makeDatabase.R” after running the code.

➢ This script produces 3 additional files with the same name as the file used to create the database. These files have the terminations: “nhr”, “nin” and “nsq”.

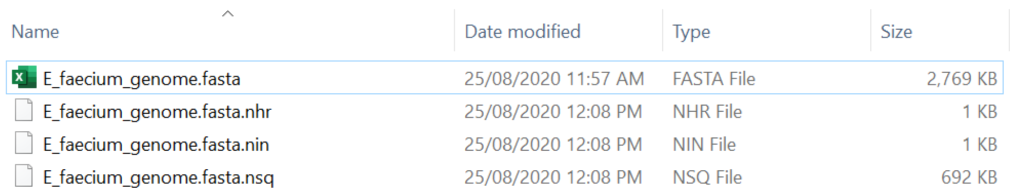


Figure 24 Example of files created with the code "makeDatabase.R".

4. **BLAST and filtering.**
   - Select the script "BLASTnFilter.R".
   - Clean your global environment before running this script.

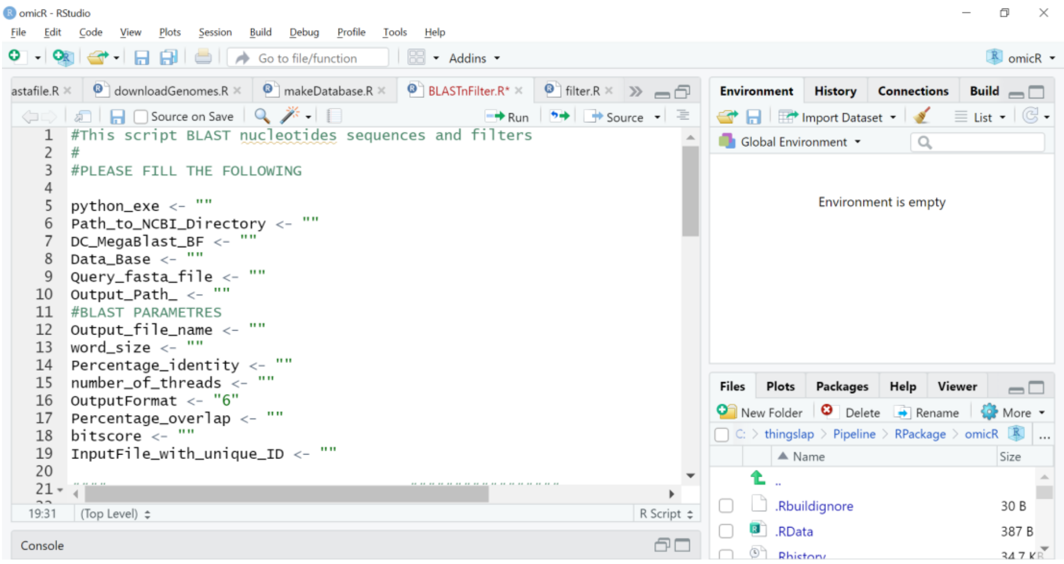


Figure 25 Example of window showing BLASTnFilter.R script.

- Fill the script and add the path to the Python executable, path to the BLAST+ bin directory, write TRUE if you are running a Discontinuous mega BLAST or FALSE if you are not, path to the BLAST+ database, path to the fasta file query, output path, output name, word size, percentage identity, number of threads, output format (only tabular format 6 is accepted), percentage overlap, bitscore, and path to file with Unique ID (Created in step 1).

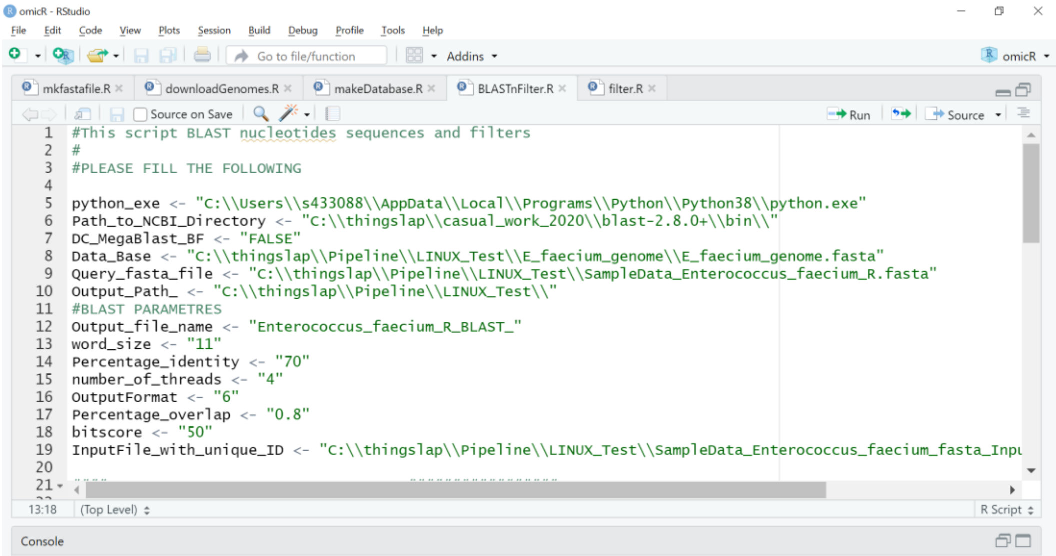


Figure 26 Example of script BLASTnFilter.R after filling parameters.

- Save the script, select all the code with CRTL+A, and run the code.

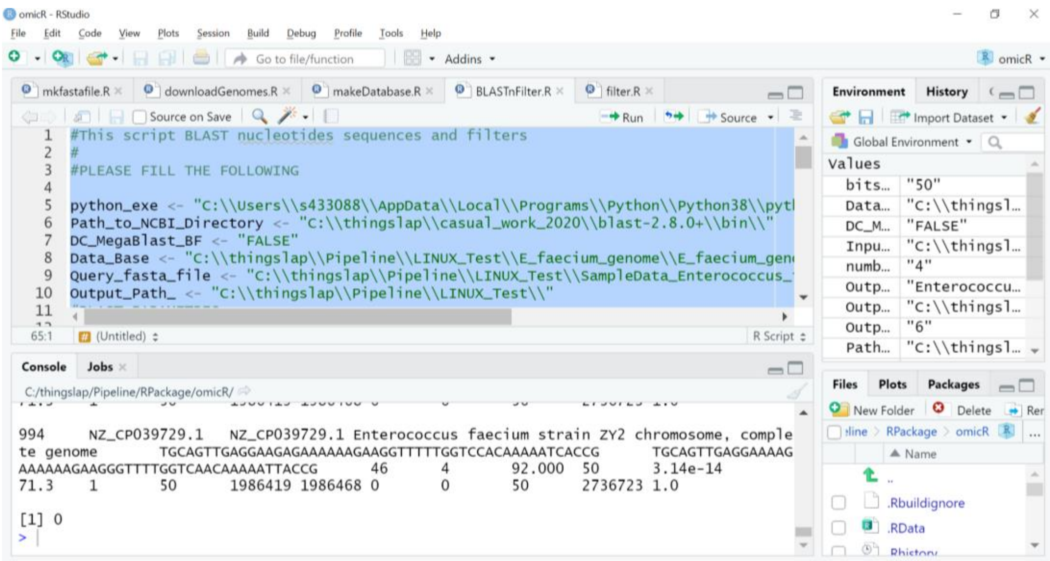


Figure 27 Example of script BLASTnFilter.R after running the code.

This script creates 5 files.

Name	Date modified	Type	Size
Enterococcus_faecium_R_BLAST__all_sequences_with_and_without_hits.txt	25/08/2020 12:47 PM	Text Document	425 KB
Enterococcus_faecium_R_BLAST__filtered.txt	25/08/2020 12:47 PM	Text Document	257 KB
Enterococcus_faecium_R_BLAST__only_sequences_with_hits.txt	25/08/2020 12:47 PM	Text Document	377 KB
Enterococcus_faecium_R_BLAST__sorted.txt	25/08/2020 12:47 PM	Text Document	215 KB
Enterococcus_faecium_R_BLAST_BLAST.txt	25/08/2020 12:47 PM	Text Document	3,057 KB

Figure 28 Example of files created by BLASTnFilter.R script.

This step takes less than 5 minutes. The script produces 5 files if you included all inputs, or 3 files if you did not provide the File with UniqueID.

Description of files produced.

- **File 1. Enterococcus_faecium_R_BLAST_BLAST.txt**
  This is the raw BLAST output. This file does not contain any headers and it is not filtered.

- **File 2. Enterococcus_faecium_R_BLAST__filtered.txt**
  This file has headers and an extra column with the percentage overlap. Default filtering parameters for this tutorial are: Percentage Overlap >80%, bitscore >50, Percentage Identity>70. The percentage overlap can be modified according to the BLAST results expected. This file may contain multiple hits per sequence.

  The BLAST has a table format 6 in the following order: "qseqid sacc stitle qseq sseq nident mismatch pident length evalue bitscore qstart qend sstart send gapopen gaps qlen slen". These are:

  - qseqid: query (e.g., unknown gene) sequence id
  - sacc: Subject accession
  - stitle: Subject Title
  - qseq: Aligned part of query sequence
  - sseq: Aligned part of subject sequence
  - nident: Number of identical matches
  - mismatch: number of mismatches
  - pident: percentage of identical matches
  - length: alignment length (sequence overlap)
  - evalue: expect value
  - bitscore: bit score
  - qstart: start of alignment in query
  - qend: end of alignment in query
  - sstart: start of alignment in subject
  - send: end of alignment in subject
  - gapopen: number of gap openings
  - gaps: Total number of gaps
  - qlen: Query sequence length
  - slen: Subject sequence length

- **File 3. Enterococcus_faecium_R_BLAST__sorted.txt**
  This file contains only one hit per sequence. The best match will be selected considering levels of sorting. First considering the highest percentage identity, then highest Percentage overlap, then highest bitscore and then taking only one Query per sequence.
  If you didn't provide the file with UniqueID, this will be your final result.

- **File 4. Enterococcus_faecium_R_BLAST__only_sequences_with_hits.txt**
  This file uses the UniqueID assigned to each sequence and writes the BLAST results back into the original file. This file only contains sequences that had a hit to something.

- **File 5. Enterococcus_faecium_R_BLAST__all_sequences_with_and_without_hits.txt**
  This file uses the UniqueID assigned to each sequence and writes the BLAST results back into the original file. This file only contains all sequences with and without hits, written back into the original file.

5. Running BLAST and filtering script without the file with Unique ID.

If you do not have the file with Unique ID created with this pipeline, you can still run the BLAST and filtering. The steps are the same for running a normal BLAST but leave empty the string "InputFile_with_unique_ID" as shown in the picture.

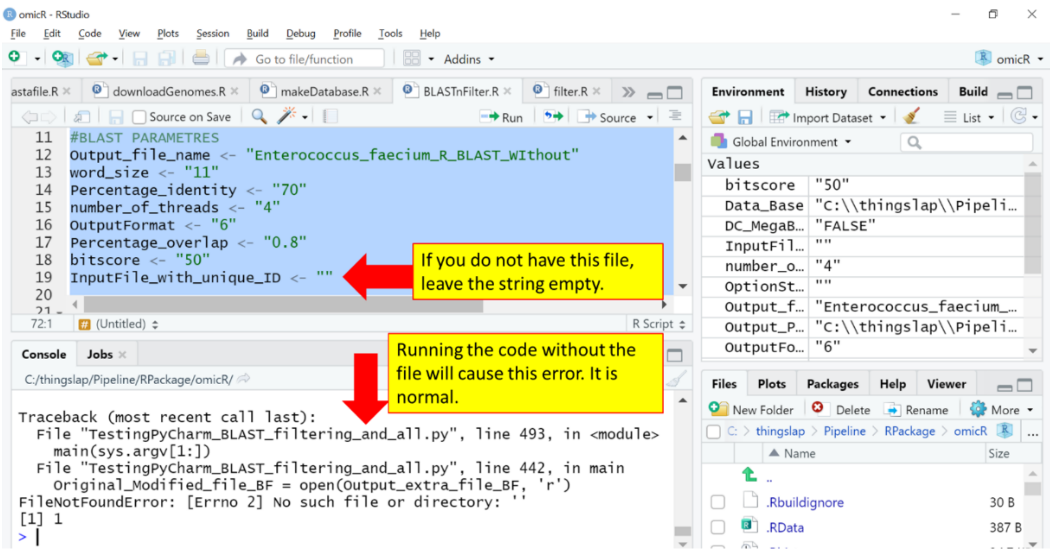


Figure 29 Example of script BLASTnFilter.R after running the code without the file with UniqueID.

Running the code without this file will produce 4 text documents. Three of them with data and the final one empty. Your final file with the sequences of interest will have the termination "_sorted.txt".

Name	Date modified	Type	Size
Enterococcus_faecium_R_BLAST_WIthout_all_sequences_with_and_without_hits.txt	25/08/2020 2:16 PM	Text Document	0 KB
Enterococcus_faecium_R_BLAST_WIthout_sorted.txt	25/08/2020 2:16 PM	Text Document	215 KB
Enterococcus_faecium_R_BLAST_WIthout_filtered.txt	25/08/2020 2:16 PM	Text Document	257 KB
Enterococcus_faecium_R_BLAST_WIthoutBLAST.txt	25/08/2020 2:16 PM	Text Document	3,057 KB

Figure 30 Example output files generated after running the script BLASTnFilter.R without the file with UniqueID.

6. **Additional filtering.**

- For further filtering with different parameters without running BLAST+ again, you can use the script “filter.R”.
- Clean the global environment before you run this script.

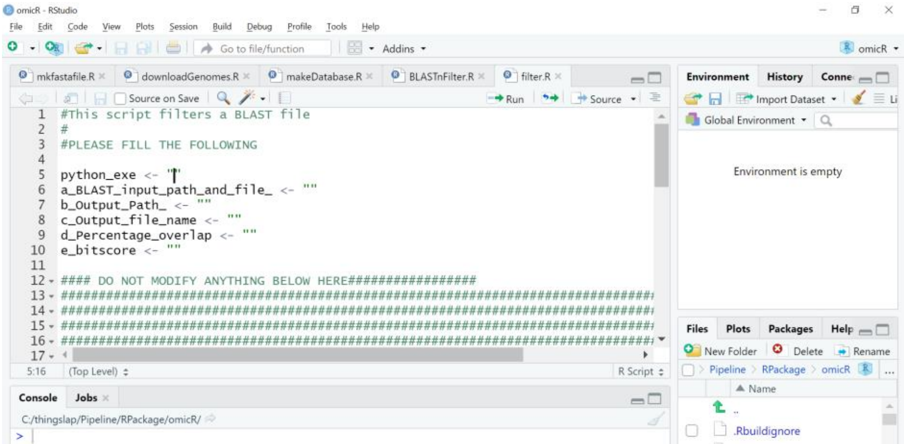


Figure 31 Example of filter.R script.

- Fill the script and add the path to the Python executable, path to the BLAST output file produced in the previous step, output path, output name, percentage overlap and bitscore.
- This script can work on any BLAST output file which has exactly the same format as is used here. The format used here is BLAST output format 6, with the following columns: qseqid sacc stitle qseq sseq nident mismatch pident length evalue bitscore qstart qend sstart send gapopen gaps qlen slen.

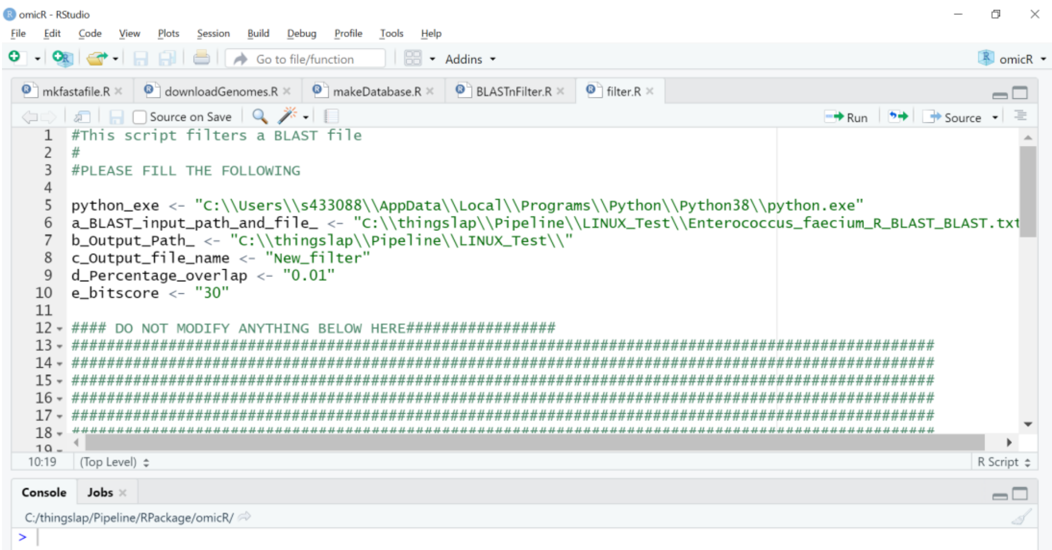


Figure 32 Example of "filter.R script" after filling parameters.

- Save the script, select all the code with CRTL+A, and run the code.

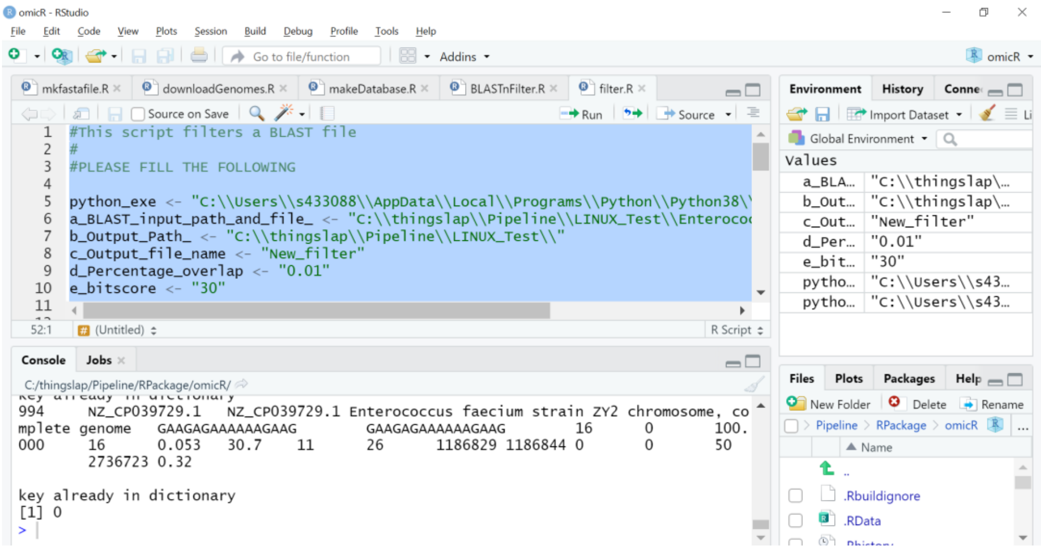


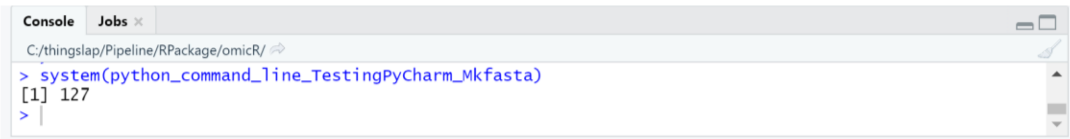
Figure 33 Example of "filter.R script" after running the code.

This will produce 2 files, with the new filtering parameters selected. The file with the name you provided and the ending "_sorted.txt" document is your final file.

New_filter_filtered.txt	25/08/2020 2:42 PM	Text Document	424 KB
New_filter_sorted.txt	25/08/2020 2:42 PM	Text Document	240 KB

7. Common errors

- If R is running, not producing any script and giving this error [1] 127, it means that the compiler is not found. Check that you are selecting the correct path to the Python executable and writing \\ in the path.



- Writing paths with a blank space between words.
- Selecting the wrong file or wrong path.
- Selecting "excel files" instead of "csv" files as input for initial step.
- Not adding "\\" at the end of each path to a directory.

omicR for Linux

Download

https://github.com/BTalamantesBecerra/omicR_for_linux

Requirements

- NCBI BLAST+ V4 or latest. (https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/)
- Python V3 or latest (https://www.python.org/downloads/)
    - Package "tkinter"
- Biopython (https://biopython.org/)
    - Package "entrez"

Note: The files "omicR.py" and "Currito.xbm" need to be in the same directory.

Running instructions

Alternative 1:

-Open the script "omicR.py" using the IDLE or PyCharm and run it.

-This will open a window showing the Graphical User Interface where you can run the scripts.

Alternative 2:

-Open the script from the command line. Move directories to the location where you downloaded the script and run:

```
python3 omicR.py
```

- This will open a window showing the Graphical User Interface where you can run the scripts.

For usage, please refer to the section "Running "omicR" with graphical user interface" in this manual.

omicR for HPC computers

If you are using Windows, I recommend downloading and installing Putty and WinSCP.

Download

https://github.com/BTalamantesBecerra/omicR_linux_commandline

Requirements

- NCBI BLAST+ V4 or latest. (https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/)
- Python V3 or latest (https://www.python.org/downloads/)
- Biopython (https://biopython.org/)

Add these programs to your environment path variables.

Introduction

If you are running omicR with an HPC computer, it likely that you know how to use a command line. For this purpose, I suggest that you only use 2 scripts to “create fasta files” and “filter”. As the steps of downloading, creating a database and running BLAST can take longer than running BLAST+ directly.

The required input BLAST command line to run this filtering script is:

blastn -db [ ] -query [ ] -out [ ] -word_size [ ] -perc_identity [ ] -num_threads [ ] -outfmt ' 6 qseqid sacc stitle qseq sseq nident mismatch pident length evalue bitscore qstart qend sstart send gapopen gaps qlen slen’

In the following section, I will describe the steps for running all scripts.

STEPS

1. **Create FASTA files and input files for BLAST / filtering.**

Structure of the command line.

python3 (Call Python)
TestingPyCharm_Mkfasta.py (Name of the Python script)
-a (Path and name of Input CSV File)
-b (Output path)
-c (Name of output file)
-d (Row with header- start counting from 1)
-e (Row with data- start counting from 1)

-f (Column with sequence of interest - start counting from 0)

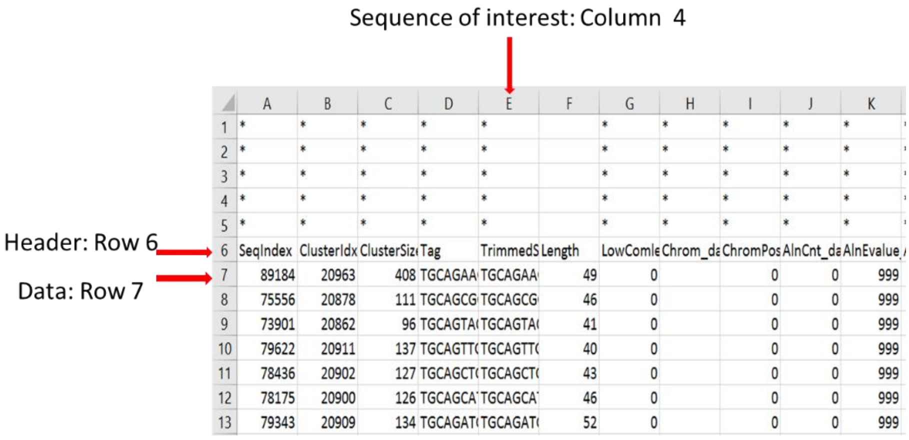


Figure 34 Display of how rows and headers should be selected.

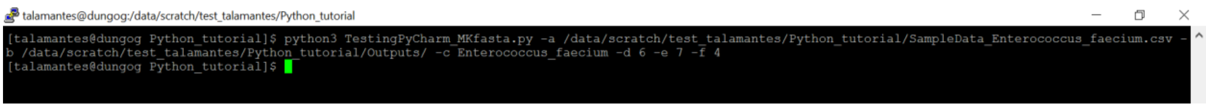
An example of how the command line should look is below:

```
Python3 TestingPyCharm_Mkfasta.py -a ~Path\YourFile.csv -b ~Path\Directory\ -c YourFileName -d 7 -e 8 -f 2
```

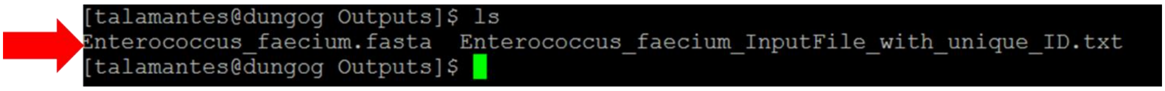
Note: Before running this command line, you must be in the location where the script is saved to be able to run it.

Move directories until you get to the location where the Python script was saved.

This is an example of how it should look



The two output files should be in the Outputs directory:



2. **Download genomes from the NCBI.**

Structure of the command line.

python3 (Call Python)

TestingPyCharm_Downloading_genomes.py (Name of the Python script)

-a (your email)

-b (RefSeq number)

-c (Path for outputs)

-d (Name of output file)

An example of how the command line should look is given below:

```
Python3 TestingPyCharm_Downloading_genomes.py -a
youremail@hotmail.com -b GCF_900067755.1 -c ~Path\Directory\ -d
YourGenomeName
```

This script is recommended for small genomes or chromosomes. If you are planning to download large genomes from NCBI, it is recommended to use the browser to download from the NCBI website. You cannot modify the speed of fetching in this step. For example, a bacterial genome such as NZ_CP039730.1 takes a few seconds, but the entire chicken genome can take approximately 5 hours.

You need to provide your email as NCBI needs to identify you for granting you access, otherwise your access can be denied.

Note: Before running this command line, you must be in the location where the script is saved to be able to run it.

Move directories until you get to the location where the Python script was saved.

3. Create NCBI database for BLAST+

Structure of the command line.

python3 (Call Python)

TestingPyCharm_MakeDataBase.py (Name of the Python script)

-a Path to your BLAST+ bin directory

-b (Path to your genome file as fasta of fna file)

-c (Type of database, use nucl for nucleotides)

An example of how the command line should look is given below:

python3 TestingPyCharm_MakeDataBase.py -a ~PATH\blast-2.8.0+\bin\ -b ~Path\MyGenome.fna -c nucl

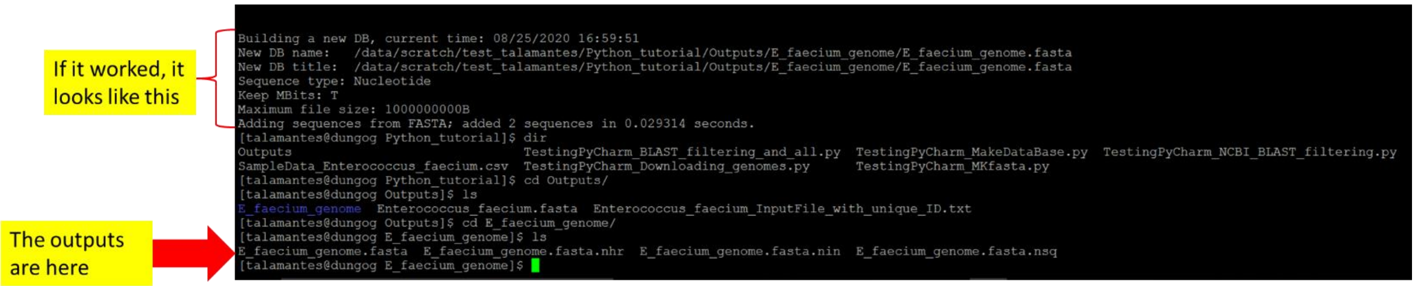
Notes:

*** Before running this command line, you must be in the location where the script is saved to be able to run it.**

***Move directories until you get to the location where the Python script was saved.**

***If you already have the mkblastdb in the global environment paths do not include the handle '-a'**

If it works, it should look like this:



4. BLAST and filtering.

Structure of the command line:

python3 (Call Python)

TestingPyCharm_BLAST_filtering_and_all.py (Name of the Python script)

-y (**TRUE** if you are running a Dissimilar discontinuous MegaBLAST or **FALSE** if you are not)

-a (Path to the reference genome)

-b (Path to the query sequences)

-c (Path to the output directory)

-d (Name of output directory)

-e (Word Size: recommended value 11)

-f (Percentage identity: recommended value 70)

-g (Threads)

-i (Output format, only tabular format 6 is accepted for this script)

-j (Percentage overlap: recommended value 0.8)

-k (Bitscore: recommended value 50)

An example of how the command line should look like is below:

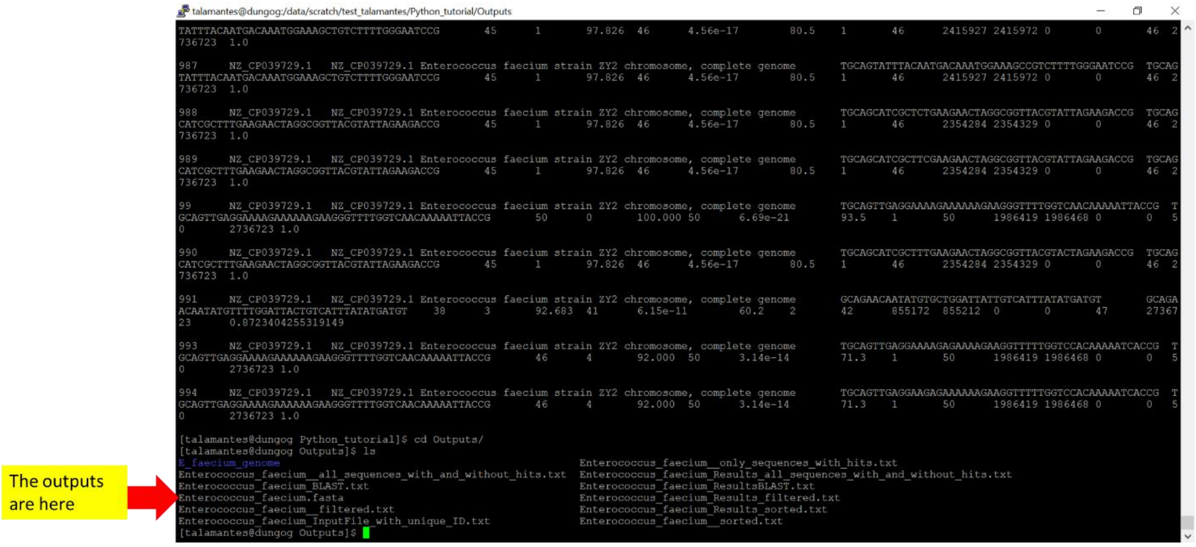
```
python3 TestingPyCharm_BLAST_filtering_and_all.py -y FALSE -a
Path/MyGenome.fna -b Path/MySequences.fasta -c ~Path/directory/ -d
MyBlastResults -e 11 -f 70 -g 35 -i 6 -j 0.8 -k 50 -l
```

Notes: Before running this command line, you must be in the location where the script is saved to be able to run it.

***Move directories until you get to the location where the Python script was saved.**

If you ran the command line without including the file with unique ID, one file without data will be generated "Enterococcus_faecium_Results_all_sequences_with_and_without_hits.txt". This is normal. Your final file has the name "Enterococcus_faecium_Results_sorted.txt"

If you ran these two command lines, your outputs will be similar to this:



It will produce the 5 files, or the 3 files if you did not provide the File with UniqueID.

Description of files produced.

- **File 1. Enterococcus_faecium_BLAST.txt**
  This is the raw BLAST output. This file does not contain any headers and it is not filtered.

- **File 2. Enterococcus_faecium_filtered.txt**
  This file has headers and an extra column with the percentage overlap. Default filtering parameters for this tutorial are: Percentage Overlap >80%, bitscore >50, Percentage Identity>70. The percentage overlap can be modified according to the BLAST results expected. This file may contain multiple hits per sequence.

  The BLAST has a table format 6 in the following order: "qseqid sacc stitle qseq sseq nident mismatch pident length evalue bitscore qstart qend sstart send gapopen gaps qlen slen". These are:

  - qseqid: query (e.g., unknown gene) sequence id
  - sacc: Subject accession
  - stitle: Subject Title
  - qseq: Aligned part of query sequence
  - sseq: Aligned part of subject sequence
  - nident: Number of identical matches
  - mismatch: number of mismatches
  - pident: percentage of identical matches
  - length: alignment length (sequence overlap)
  - evalue: expect value
  - bitscore: bit score
  - qstart: start of alignment in query

- qend: end of alignment in query
- sstart: start of alignment in subject
- send: end of alignment in subject
- gapopen: number of gap openings
- gaps: Total number of gaps
- qlen: Query sequence length
- slen: Subject sequence length

- **File 3. Enterococcus_faecium_sorted.txt**
  This file contains only one hit per sequence. The best match will be selected considering levels of sorting. First considering the highest percentage identity, then highest Percentage overlap, then highest bitscore and then taking only one Query per sequence.
  If you didn't provide the file with UniqueID, this will be your final result.

- **File 4. Enterococcus_faecium_only_sequences_with_hits.txt**
  This file uses the UniqueID assigned to each sequence and writes the BLAST results back into the original file. This file only contains sequences that had a hit to something.

- **File 5. Enterococcus_faecium_all_sequences_with_and_without_hits.txt**
  This file uses the UniqueID assigned to each sequence and writes the BLAST results back into the original file. This file only contains all sequences with and without hits, written back into the original file.

5. **Additional filtering.**

Structure of the command line:

python3 (Call Python)

TestingPyCharm_NCBI_BLAST_filtering.py (Name of the Python script)

-a (Path and name of BLAST output file)

-b (Path to the output directory)

-c (Name of output directory)

-d (Percentage overlap: recommended value 0.8)

-e (Bitscore: recommended value 50)

An example of how the command line should look like is below:

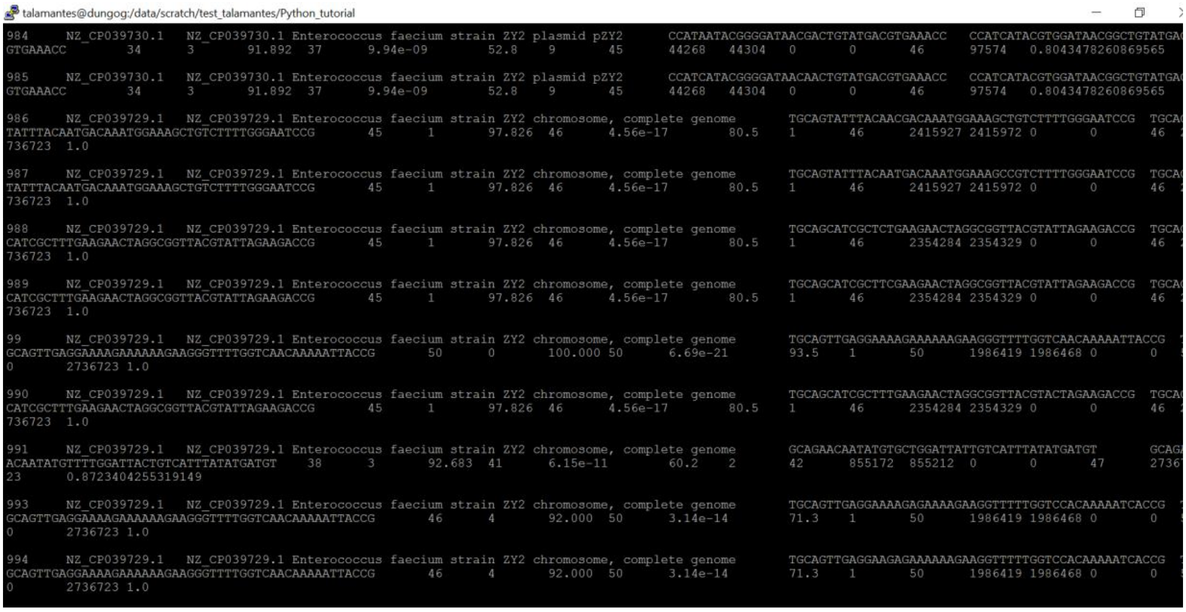
```
python3 TestingPyCharm_NCBI_BLAST_filtering.py -a Path/MyBLAST_Results.txt
-b ~Path/directory/ -c My_Filtered_Blast_Results -d 0.8 -e 50
```

Note: Before running this command line, you must be in the location where the script is saved to be able to run it.

Move directories until you get to the location where the Python script was saved.

***You have to run the previous step (or command lines) to be able to run this command line.**

If you run it, the screen will look like this.



The two new files will be in the output directory with the other files.

The End